同步异步

练习1

```
<script>
        console.log(1, 'script start') //  1
        setTimeout(function () {
            console.log(3, 'settimeout') //  3
        })
        console.log(2, 'script end') //  2
    </script>
```

练习2

```
console.log(1, 'script start');//

        let promise1 = new Promise(function (resolve) {
            console.log(2, 'promise1');//
            resolve();
            console.log(3, 'promise1 end');//
        }).then(function () {
            console.log(5, 'promise2');//
        });

        setTimeout(function () {
            console.log(6, 'settimeout')//
        });

        console.log(4, 'script end');//
```

练习3

```
async function async1() {
            console.log(2, 'async1 start');//
            await async2();
            console.log(5, 'async1 end')//
        }

        async function async2() {
            console.log(3, 'async2')//
        }

        console.log(1, 'script start');//

        async1();

        console.log(4, 'script end')//
```

练习4

```
async function async1() {
            console.log(2, 'async1 start'); //2
            await async2();
            console.log(6, 'async1 end'); //6
        }

        async function async2() {
            console.log(3, 'async2'); //3
        }

        console.log(1, 'script start'); //1

        setTimeout(function () {
            console.log(8, 'setTimeout'); //8
        }, 0)

        async1();

        new Promise(function (resolve) {
            console.log(4, 'promise1'); //4
            resolve();
        }).then(function () {
            console.log(7, 'promise2'); //7
        });

        console.log(5, 'script end'); //5
```

面试题：

题1

```
//给函数参数设置默认值的时候，需要设置到尾参数身上
//参数是没有顺序要求，在函数内部任何地方都可以使用
//形参是函数内部的全局变量
   function fn(b, c, a = 10) {
        console.log(b, c, a);
        return b + a + c;
    }
    fn(2, 3)
```

题2

```
          function fun(n, o) {
             console.log(o);
             return {
                 fun: function (m) {
                     return fun(m, n);
                 }
             };
         }

         // 1--------------------------------------------------------------------------
         var a = fun(0); //1 undefined --->形参没有接受到实参
         /*********
         分析： AO-->  n:0
             a: {
                 fun: function(m) {
                     return fun (m, n);
                 }
             }
         */
         a.fun(1); //2 o--->n--->0
         /** 分析：
                1-->m --->return fun(1, 0);
         */
```

```
        a.fun(2); //3 o--->n--->0
        /** 分析:
               2-->m --->return fun(2, 0);
        */
        a.fun(3); //4 o--->n--->0
        a.fun(32227878787882);

        // 2-----------------------------------------------------------------------
        var b = fun(0).fun(1).fun(2).fun(3).fun(500).fun(0); //undefined-->0-->1-->2-->500
        /*
        分析: fun(0): AO-->  n:0
            b: {
               fun: function(m) {
                  return fun (m, n);
               }
            }
        */

        // 3-----------------------------------------------------------------------
        var c = fun(0).fun(1); //undefined  0
        /**
        c 这个对象里面的n-->1
        c: {
            fun: function (m) {
                return fun(m, n);
            }
        }
        */
        c.fun(2); //  1
        c = c.fun(3); //1-->3
        c.fun(999); //3
        c.fun(123); //3
```

题3：数组去重

```
         let arr1 = [1, 2, 3, 2, 1];

        //1
        function p1(arr) {
            let newArr = [];
            for (let i = 0; i < arr.length; i++) {
                newArr.indexOf(arr[i]) === -1 && newArr.push(arr[i]);
            }
            return newArr;
        }

        //2
        function p2(arr) {
            return Array.from(new Set(arr));
        }

        //3
        function p3(arr) {
            return [...new Set(arr)];
        }

        //4
        function p4(arr) {
            return arr.filter((item, ind) => arr.indexOf(item) === ind);
        }

        console.log(p1(arr1));//[1, 2, 3]
        console.log(p2(arr1));//[1, 2, 3]
        console.log(p3(arr1));//[1, 2, 3]
        console.log(p4(arr1));//[1, 2, 3]

        //reduce
        var arr = [1, 2, 3, 4, 5, 6, 4, 3, 8, 1];
        // 数组去重:
        // 方法12 : reduce
        function newArrFn(arr) {
            let newArr = [];
            return arr.reduce((prev, next, index, arr) => {
                // 如果包含，就返回原数据，不包含，就把新数据追加进去
                return newArr.includes(next) ? newArr : newArr.push(next);
            }, 0)
        }
        console.log(newArrFn(arr));
```

```

         //数组的indexOf方法：遍历数组，找到第一个传入的参数，如果没有就返回-1
        let arr = [1, 2, 2, 2, 34, 4, 34, 5];

        function fn(arr) {
            let newArr = [];
            for (let i = 0; i < arr.length; i++) {
                if (newArr.indexOf(arr[i]) === -1) {
                    newArr.push(arr[i]);
                }
            }
            return newArr;
        }
        console.log(fn(arr)); //[1, 2, 34, 4, 5]

         //定义一个新数组，放入原数组第一项，双重for循环遍历对比每一项，没有重复的就加入新数组：
        let arr = [1, 2, 2, 2, 343, 4, 34, 5];

        function fn(arr) {
            let newArr = [arr[0]];
            for (let i = 0; i < arr.length; i++) {
                let flag = false;
                for (let j = 0; j < newArr.length; j++) {
                    if (arr[i] === newArr[j]) {
                        flag = true;
                        break;
                    }
                }
                if (!flag) {
                    newArr.push(arr[i]);
                }
            }
            return newArr;
        }
        console.log(fn(arr)); //[1, 2, 343, 4, 34, 5]

        //数组的includes方法用于判断判断一个数组是否包含一个指定的值。如果是则返回 true，否则返回 false。
        let arr = [1, 2, 2, 2, 343, 4, 34, 5];
        function fn(arr) {
            let newArr = [];
            for (let i = 0; i < arr.length; i++) {
                if (!newArr.includes(arr[i])) {
                    newArr.push(arr[i]);
                }
            }
            return newArr;
        }
        console.log(fn(arr)); //[1, 2, 343, 4, 34, 5]

         //filter方法是根据过滤条件对整个数组进行过滤，符合的为true保留，不符合的为false删除。
        let arr = [1, 2, 2, 2, 343, 4, 34, 5];

        let arr2 = arr.filter(function (item, ind) {
            return arr.indexOf(item) === ind;
        });
        console.log(arr2);

         //ES6的set方法去重
        let arr = [1, 2, 2, 2, 3, 4, 34, 5];
        let arr1 = [1, 2, , 2, 4, 34, 5]

        let arr2 = [...new Set(arr)];
        let arr3 = [...new Set(arr, arr1)];

        console.log(arr2); // [1, 2, 343, 4, 34, 5]
        console.log(arr3);

        //当数组里含有复杂数据类型的时候，一般的去重方法就没用了，
        //但是可以用filter和hasOwnProperty方法去重
        let arr = [1, 1, 2, 2, 3, 3, {}, {}, null, null, NaN, NaN, [],
            [], '', '', false, false, true, true
        ];
        let arr1 = [];
        arr1 = arr.filter(function (item) {
            if (!arr.hasOwnProperty(typeof item + item)) {
                return arr[typeof item + item] = true;
```

```javascript
            }
        });
        console.log(arr1);
        //[1, 2, 3, {}, null, NaN, [], '', false, true]

        //遍历数组，利用数组的indexOf方法判断此元素在该数组中首次出现的索引
        //与循环的i是否相等;再利用splice方法删除不符合条件的元素,并及时调整i;
        let arr = [1, 2, 3, 4, 5, 3, 2, 3, 4, 1, 11, 22, 2, 5, 6];

        function myArr(arr) {
            for (let i = 0; i < arr.length; i++) {
                if (arr.indexOf(arr[i]) != i) {
                    arr.splice(i, 1);
                    i--;
                }
            }
            return arr;
        }
        console.log(myArr(arr));
        //[1, 2, 3, 4, 5, 11, 22, 6]

         //主要利用findIndex 的特性，查找元素找不到就返回-1， 接下来
        //就需要判断，如果是-1，说明没找到，就往新数组里面添加元素。
        let arr = [1, 2, 3, 4, 5, 3, 2, 3, 4, 1, 11, 22, 2, 5, 6];

        function myArr(arr) {
            let newArr = [];
            for (let i = 0; i < arr.length; i++) {
                newArr.indexOf(arr[i]) === -1 ? newArr.push(arr[i]) : newArr;
            }
            return newArr;
        }
        console.log(myArr(arr));
        //[1, 2, 3, 4, 5, 11, 22, 6]

        //首先利用 sort 方法进行排序。进行循环，如果原数组的第 i 项和新数组的 i - 1 项不一致，就push进去。
        let arr = [1, 2, 3, 4, 5, 3, 2, 3, 4, 1, 11, 22, 2, 5, 6];

        function myArr(arr) {
            arr = arr.sort();
            let newArr = [];
            for (let i = 0; i < arr.length; i++) {
                arr[i] === arr[i - 1] ? newArr : newArr.push(arr[i]);

            }
            return newArr;
        }
        console.log(myArr(arr)); //[1, 11, 2, 22, 3, 4, 5, 6]

        //利用 set数据不重复的特点，结合 Array.from
        let arr = [1, 2, 3, 4, 5, 3, 2, 3, 4, 1, 11, 22, 2, 5, 6];

        function newArrFn(arr) {
            // .new Set方法，返回是一个类数组，需要结合 Array.from ，转成真实数组
            return (Array.from(new Set(arr)))
        }
        console.log(newArrFn(arr));

        //利用数据结构存值的特点
        var arr = [1, 2, 3, 4, 5, 6, 4, 3, 8, 1]
        // 数组去重：
        // 方法11 : Map
        function newArrFn(arr) {
            let newArr = []
            let map = new Map()
            for (let i = 0; i < arr.length; i++) {
                // 如果 map里面不包含，就设置进去
                if (!map.has(arr[i])) {
                    map.set(arr[i], true)
                    newArr.push(arr[i])
                }
            };
            return newArr
        }
        console.log(newArrFn(arr));
```

题4：数组拷贝

```
let arr1 = [1, 2, 3, {
            a: 1,
            b: [3, 4, 5]
        }, undefined, [5, 6]];
        /*
        function copyArray(arr) {
            let newArray = [];
            //循环
            for (let j = 0; j < arr.length; j++)newArray.push(arr[j]);
            return newArray;
        }

        function copyArray(arr) {
            return [...arr];
        }

        function copyArray(arr) {
            return JSON.parse(JSON.stringify(arr));
        }
        */

        //浅拷贝-->深拷贝
        let arr2 = [1, 2, 3, {
            a: 1,
            b: [3, 4, 5]
        }, undefined, [5, 6]];

        function copyArray(arr) {
            if (arr instanceof Array) {
                let newArray = [];
                for (let j = 0; j < arr.length; j++) newArray.push(copyArray(arr[j]));
                return newArray;
            } else if (arr instanceof Object) {
                let buffer = {};
                for (let k in arr) buffer[k] = copyArray(arr[k]);
                return buffer;
            } else {
                //找到边界墙  临界点
                return arr;
                }
            }
            console.log(copyArray(arr1), copyArray(arr1) === arr1);
            console.log(copyArray(arr1)[5] === arr1[5]); //false
            console.log(copyArray(arr1)[3].b === arr1[3].b); //false
            console.log(copyArray(arr2));//[1, 2, 3, {…}, undefined, Array(2)]
```

题5：回文

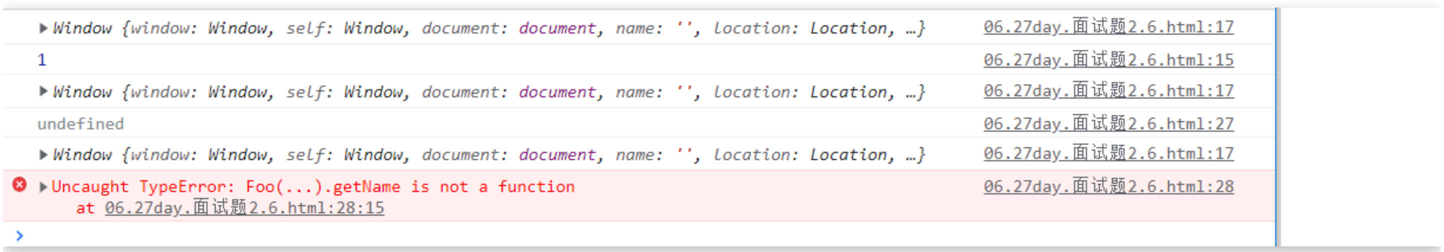
```
let str1 = '121';
        function reverStr(str) {
            str = str.toString();
            return str === str.split('').reverse().join('');
        }
        console.log(reverStr(str1));
        console.log(reverStr('str1'));
        console.log(reverStr('666'));
```

题6：函数调用

```
function Foo() {
            getName = function () {
                console.log(1);
            };
            console.log(this);
            return this;
        }

        let getName = function () {
            console.log(4);
        };

        Foo();
        getName(); //1
        console.log(Foo().getName);
        Foo().getName();
```



```
▶Window {window: Window, self: Window, document: document, name: '', location: Location, …}    06.27day.面试题2.6.html:17
1                                                                                               06.27day.面试题2.6.html:15
▶Window {window: Window, self: Window, document: document, name: '', location: Location, …}    06.27day.面试题2.6.html:17
undefined                                                                                       06.27day.面试题2.6.html:27
▶Window {window: Window, self: Window, document: document, name: '', location: Location, …}    06.27day.面试题2.6.html:17
❌ ▶Uncaught TypeError: Foo(...).getName is not a function                                       06.27day.面试题2.6.html:28
     at 06.27day.面试题2.6.html:28:15
>
```

```javascript
var getName = function () {
            console.log(4);
        };

        function getName() {
            console.log(5);
        }
        getName(); //4
```

题7：原型链查找

```javascript
function foo() {
            //直接修改静态方法
            foo.a = function () { console.log(1) };
            //给实例化追加一个方法 a
            this.a = function () { console.log(2) };
            //局部变量
            a = function () { console.log(3) };
            var a = function () { console.log(4) };
        }
        foo.prototype.a = function () { console.log(5) };
        foo.a = function () { console.log(6) };//class  static  静态方法
        foo.a(); //6

        var obj = new foo();//构造函数里面的代码是从头开始执行
        console.log(obj);
        //优先找自己，如果没有找到就顺着原型一层一层往上找，直到找到为止
        obj.a(); //2
        foo.a(); //1
```

题8：统计字符出现次数

```
//'use strict';
        var str = 'asdfscb8sasaa';
        var json = {};
        for (var i = 0; i < str.length; i++) {
            if (!json[str.charAt(i)]) {
                json[str.charAt(i)] = 1;
            } else {
                json[str.charAt(i)]++;
            }
        };

        var iMax = 0;
        var iIndex = '';
        for (var i in json) {
            if (json[i] > iMax) {
                iMax = json[i];
                iIndex = i;
            }
        }
        console.log('出现次数最多的是:' + iIndex + '出现' + iMax + '次');
```

题9：字符串第二个单词以后首字母大写

```
//'hua-qing-yuan-jian'
        function result(str) {
            var str1 = '';

            str1 = str.toLowerCase().split('-');

            //方法1
            for (var i = 1; i < str1.length; i++) {
                str1[i] = str1[i].charAt(0).toUpperCase() + str1[i].substring(1);
            }

            //方法2
            /**
            for (var j = 1; j < str1.length; j++) {
                str1[j] = str1[j].substr(0, 1).toUpperCase() + str1[j].substr(1);
            }
            */

            //方法3
            /**
            for (var n = 1; n < str1.length; n++) {
                str1[n] = str1[n].substring(1, 0).toUpperCase() + str1[n].substring(1);
            }
            */
            return str1.join('');
        }

        var str = 'hua-qing-yuan-jian';
        console.log(result(str));
```

题10：找最大值

```
       var arr = [-1, -2, 1, 10, 4, 5, 8];
        var max1 = Math.max.apply(null, arr);
        console.log(max1);

        var max2 = arr.sort(function (a, b) {
            return b - a;
        })[0];
        console.log(max2);

        var max3 = -Infinity;
        for (var i = 0; i < arr.length; i++) {
            if (max3 < arr[i]) {
                max3 = arr[i];
            }
        }
        console.log(max3);//10
```

题12：获取时间及未来一周的时间

```
        var d = new Date();
        var today = d.getFullYear() + '-' + (d.getMonth() + 1).toString().padStart(2, '0') + '-' + d.getDate();
        console.log(today);//2022-06-12

        d.setTime(d.getTime() + 7 * 24 * 3600 * 1000);
        var nexttoday = d.getFullYear() + '-' + (d.getMonth() + 1).toString().padStart(2, '0') + '-' + d.getDate();
        console.log(nexttoday);//2022-06-19 一周后的时间

        var str = 'abcabcabcabcabcabda';
        var arr = [];
        var n = 0;
        while (str.indexOf('ab', n) != -1 && n < str.length) {
            arr.push(str.indexOf('ab', n));
            n = str.indexOf('ab', n) + 2;
        }
        console.log(arr);
        console.log(n);
```

题13：sum求和1

```
function sum(...args) { //这里的三个点...是扩展运算符，该运算符将一个数组，变为参数序列。
            if ([...args].length == 1) { //判断参数个数的形式是否为1个，即第二种形式
                var cache = [...args][0]; //将第一个参数的值暂存在cache中
                var add = function (y) { //创建一个方法用于实现第二个条件，最后并返回这个方法
                    cache += y;
                    console.log(cache)
                    return add;
                }
                return add;
            } else { //这里就是参数的第一种形式
                var res = 0; //这里最好先声明要输出的变量，并给其赋值，不然值定义而不赋值会输出NaN，因为js将undefined+number两个数据类型相加结果为NaN
                for (var i = 0; i < [...args].length; i++) {
                    res += [...args][i]; //参数累加
                }
                console.log(res) //输出最后的累加结果
            }
        }
        sum(2, 3, 4); //9
        sum(2)(3)(4)(5); //5//9//14
```

题14：sum求和2

```
var sum = (function () {
            var list = [];
            var add = function () {
                // 拼接数组
                var args = Array.prototype.slice.call(arguments);
                list = list.concat(args);
                return add;
            }
            // 覆盖 toString 方法
            add.toString = function () {
                // 计算总和
                var sum = list.reduce(function (pre, next) {
                    return pre + next;
                });
                // 清除记录
                list.length = 0;
                return sum;
            }
            return add;
        })();
        var s = sum(2, 3, 4)(3)(1, 2);
        console.log(s.toString()); //15
        console.log(sum(2, 3, 4).toString()); //9
        console.log(sum(2)(3)(4).toString()); //9
        console.log(sum(2)(3)(4).toString()); //9
        console.log(sum(1, 2)(3, 5)(4) / 1) //15
```

题15：获取字符串的字节长度，中文为2个字节

```javascript
var str = "非jnuhii";
function getStr(str) {
     var num = str.length;
     for (var i = 0; i < str.length; i++) {
/*字符串的charCodeAt()方法获取指定索引对应的ASCII码值，汉字的ASCII大于255,其它的字母数字以及其他字符ASCII编码值在0-255之间*/
          if (str.charCodeAt(i) > 255) {
              num += 1
             }
      }
      return num;
}
console.log(getStr(str));
```